Channel Decoding "Soft" Acceleration on the TigerSHARC DSP

The TigerSHARC Digital Signal Processor is a balance of many parallel resources and high bus bandwidth. The parallel resources offer high computation rates at moderate clock speeds. The high bus bandwidth enables high data rate processing. The TigerSHARC DSP architecture is an excellent candidate for applications requiring extremely high throughput, such as the channel decoding algorithms for wireless communications. Channel decoding algorithms like Viterbi and turbo decoders operate on large blocks of data and require an extremely high amount of processing per input datum. These algorithms also contain parallel characteristics allowing the processing to be split across many resources. The TigerSHARC architecture is highly suited to channel decoder processing. However, with only a few instruction set changes the TigerSHARC can excel far beyond its, and any other DSPs', current capabilities in channel decoding.

The following is a brief description of the additions to the TigerSHARC instruction set and the anticipated performance improvements for channel decoding applications (applicable to many wireless communications standards). The enhancements enable a few key features:

- maintenance 32-bit precision for accumulation of 16/8-bit inputs
- single instructions to Add, Compare and Select (ACS)
- single instruction to compare then add an error correction factor (TMAX for calculation of $\ln(e^a + e^b)$)
- flexibility to add customer IP within the decoding algorithms

Forward Error Correction Processing

Two techniques are commonly used in wireless communications to perform forward error correction (FEC). In 3G wireless systems a convolutional coding scheme is specified for voice transmission, and a parallel concatenated convolutional coding scheme (PCCC) is specified for data transmission. The convolutional encoded data is decoded using the Viterbi algorithm and the PCCC encoded data is decoded using a Turbo decoding algorithm. The Turbo and Viterbi decoding schemes are trellis based algorithms.

One form of "soft" acceleration on the TigerSHARC is to provide an instruction to perform the Butterfly computation of the Trellis. This is accomplished by adding an Add, Compare , Select (ACS) instruction. Unique to the TigerSHARC is its ability to execute multiple instructions in the same cycle. The TigerSHARC can execute multiple ACS instructions (effectively sixteen 16-bit ACS or eight 16-bit butterflies) in the same clock cycle. For both Viterbi and Turbo decoding, multiple butterflies are available for execution in parallel. The result is an efficient execution of Trellis-based algorithms by supplying the hardware to perform the largest fixed operation of the Trellis, the Butterfly.

The Turbo decoder is more complex than the Viterbi decoder, thus requiring additional considerations for "soft" acceleration. Several implementations of Turbo decoders have been published. The two most interesting due to accuracy and processing requirements are the Log MAP and the Max Log MAP. The most accurate and most desired is the Log MAP implementation, though it is more complex to implement (and requires more processing power) than the Max Log MAP implementation. There is a 0.5 dB performance difference between the two implementations and approximately a 3x computation difference with the current TigerSHARC instruction set.

The Max Log MAP uses MAX(a,b) as an estimate of $\ln(e^a + e^b)$. The estimate is derived from the Jacobian logarithm $\ln(e^a + e^b) = \text{MAX}(a,b) + \ln(1 + e^{-|a-b|})$. The Log MAP Turbo decode implementation adds the correction factor $\ln(1 + e^{-|a-b|})$ to the MAX(a,b). This produces a more accurate result. The correction factor $\ln(1 + e^{-|a-b|})$ must be implemented using a look-up table. The TMAX instruction enables a TigerSHARC implementation of the Log MAP Turbo decoder without additional cycles by adding the correction factor in the same instruction the MAX(a,b) is computed.

Many variations of these instructions are being designed into the TigerSHARC. The overall result is a high speed software implementation of the channel decoding algorithms.

New Instruction Types

Rs = TMAX(Rm, Rn)
Rs = TMAX(TRm + Rm, TRn + Rn)

Function: TMAX(a,b) = MAX(a,b) + ln(1 + $e^{-|a-b|}$) where ln(1 + $e^{-|a-b|}$) is implemented using a look up table.

Rs = ACS(Rm, Rn, Rp)

Function: The ACS function implements a Trellis butterfly. Rm represents the current metric for low order states, Rn represents the current metric for the high order states, Rp represents the local metric for the butterfly and Rs is the new metric. Low order states are states 0 to N/2-1 where N is the number of states, and the high order states are N/2 to N-1. Figure 3 depicts a Trellis butterfly.

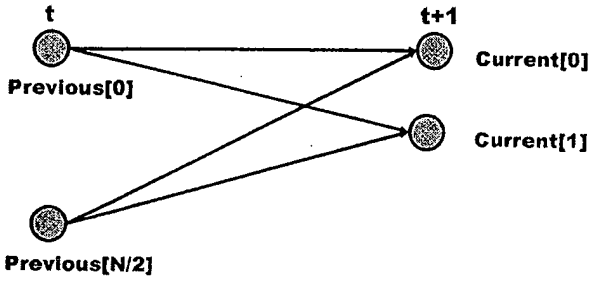

Figure 1. Trellis Butterfly

Estimated Performance Gain

The following table presents the new and the old TigerSHARC cycle requirements per channel decoder output bit. N represents the size of the block being processed. Two columns exist for the Turbo decoder performance for both the version of the TigerSHARC without "soft" acceleration (TS001) and two for the version with "soft" acceleration (TS00A). The first column represents the cycle count for the Log-MAP implementation and the second column represents the performance for the Max Log-MAP implementation. Note that with soft acceleration they are both identical.

Algorithm	TS001		TS00A	
	Cycles	Cycles	Cycles	Cycles
8-Bit Viterbi	N*91		N*67	
Metric	N*23		N*23	
Butterfly	N*52		N*28	
Scale	N*9		N*9	
Trace	N*7		N*7	
16-Bit Viterbi	N*128		N*80	
Metric	N*17		N*17	
Butterfly	N&104		N*56	
Trace	N*7		N*7	
Turbo Decoder*	N*45.50	N*15.50	N*10.5	N*10.5
Gamma	N*0.50	N*0.50	N*0.50	N*0.50
Butterfly	N*24.50	N*8.50	N*4.50	N*4.50
Extrinsic	N*20.50	N*6.50	N*5.50	N*5.50

Table 1. TigerSHARC Channel Decoder Performance Resulting from New Instructions

* Two versions of the Turbo decoder are shown. The Left column represents the Log-MAP implementation. The right column represents the Max Log-MAP implementation. The Log-MAP gives a 0.5 dB performance improvement over the Max Log-MAP